USC

Architectures for Efficient Data Mining

Zachary Baker and Viktor K. Prasanna
University of Southern California
10-11-2005

USC Viterbi
School of Engineering

Outline

- Background
  - Our work
  - FPGAs for acceleration
- Introduction to Data Mining
  - Apriori algorithm
- Architectures for the Apriori algorithm
  - Systolic array for candidate generation
  - Bitmapped CAM for faster support calculation
- Results
- Conclusion and future work

Introduction to Apriori Algorithm

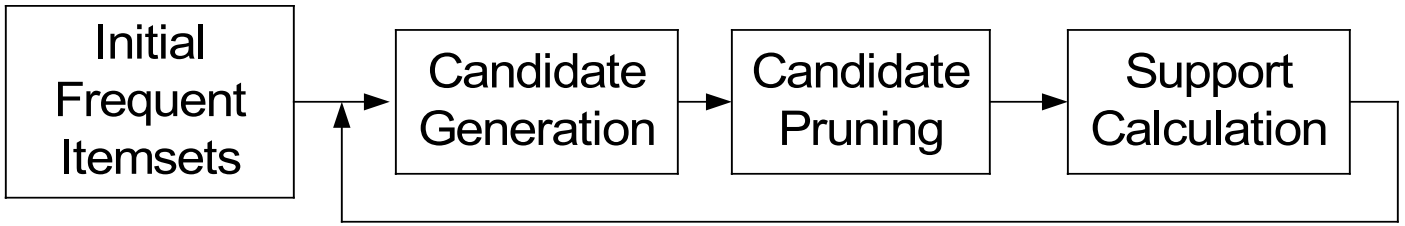
Data Mining

- Used in Intrusion Detection and other fields for inferring links between otherwise unconnected elements of large data sets
  - Packet classification: autonomous identification of attack signatures
  - Ralph's club cards
- Apriori algorithm (R. Agrawal and T. Imielinski and A. Swami, 1993)
    - Based on multiple passes of the entire dataset
      - Inferred connections are built up on each pass
      - Set operations key to the kernel

Correlation Mining Example

- Frequent singletons: bananas, cereal, milk, beer, chips
  - Frequent 2-itemsets:
    - bananas, cereal
    - bananas, milk
    - cereal, milk
    - beer, chips
  - Frequent 3-itemsets
    - bananas, cereal, milk

Elements of the Apriori Algorithm



- Initial Frequent Itemsets: common singletons
- Candidate Generation: generate new candidates by adding one element each generation
- Candidate Pruning: Remove all new candidate items that do not fulfill set requirements
- Support calculation: Stream database past candidates and determine frequency of occurrence of each candidate

Set Operations: Support

- The candidate sets are kept in local memory
- Transaction database is streamed past
  - Candidates have to see all transaction elements
  - Can require multiple passes if there are more candidates than systolic units
  - Requires much of the total time due to size of database

$\forall t \in T$ do

  $\forall c \in C$ do

    if $c \subset t$

      support(c)++

Set Operations: Candidate Gen

- Candidate generation:
  - When subset operation is satisfied, one element is appended to end of streaming set (requires injection)

$$\forall\ c_1, c_2 \in C_m \text{ do}$$
$$\text{with } c_1 = (i_1, ..., i_{m-1}, i_m)$$
$$\text{and } c_2 = (i_1, ..., i_{m-1}, i_m^*)$$
$$\text{and } i_m < i_m^*$$
$$c := c_1 \cup c_2 = (i_1, ..., i_{m-1}, i_m, i_m^*)$$

Apriori Basics

Generation 1

Item	Support
A	200
F	250
K	5
N	220
P	245

Generation 2

Item		Support
A	F	130
A	N	124
A	P	187
F	N	140
F	P	34
N	P	45

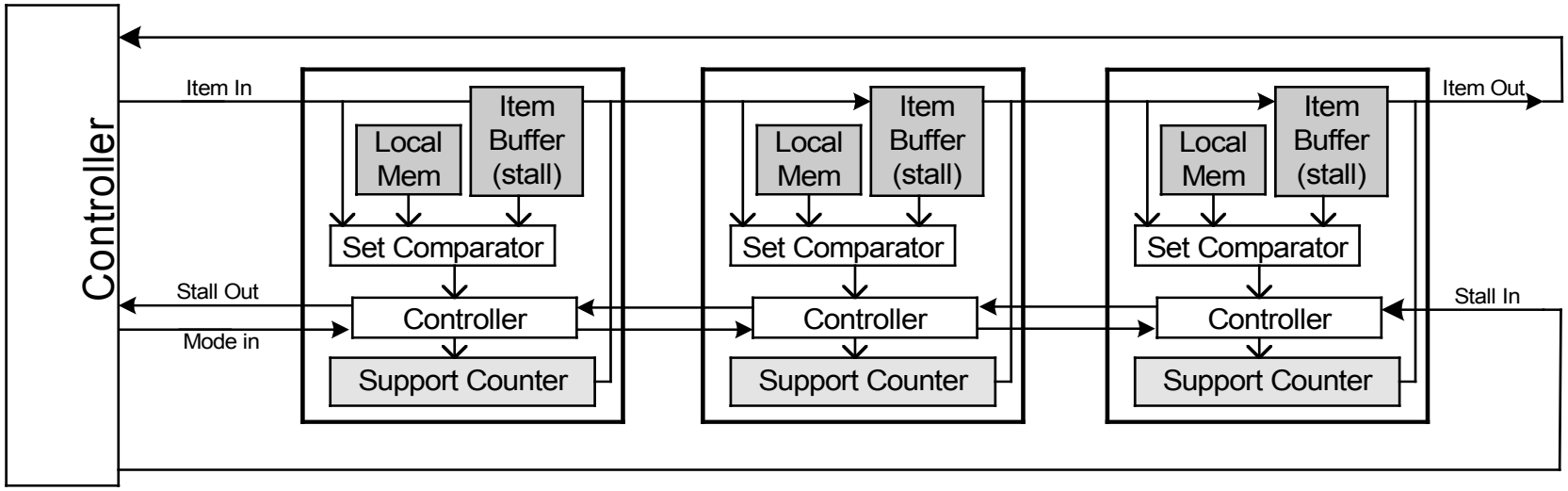
Generation 3

Item			Support
A	F	N	45
A	F	P	20
A	N	P	34

Apriori Architecture

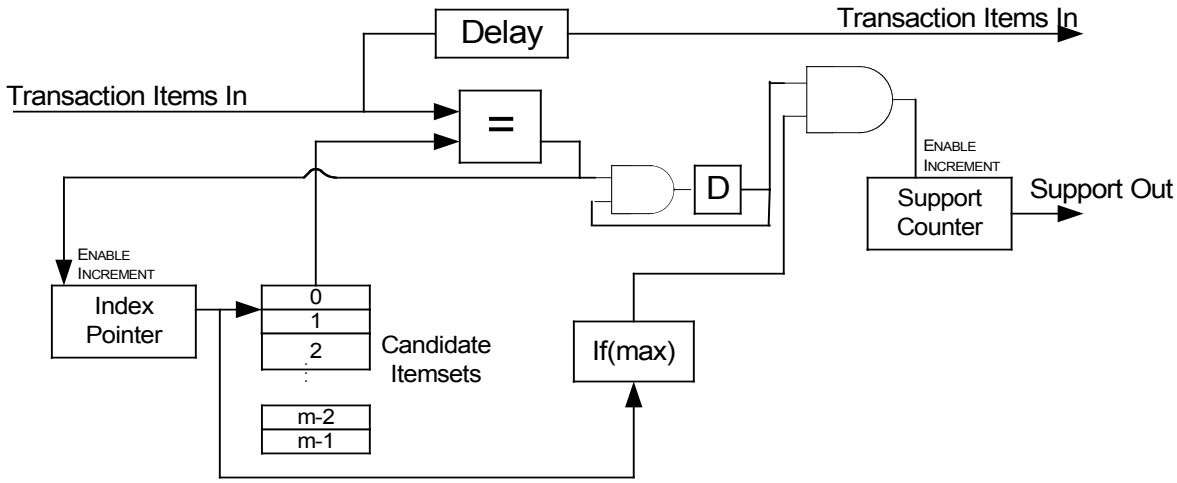
Data mining

- Set operations central to data mining kernel
- Efficiently implemented using systolic array



Set Operations: Support

- One set is kept in local memory, larger set is streamed past
  - Candidate item stored in memory
  - Reaching the end of the set memory implies subset satisfaction



Support on Sequential Machines

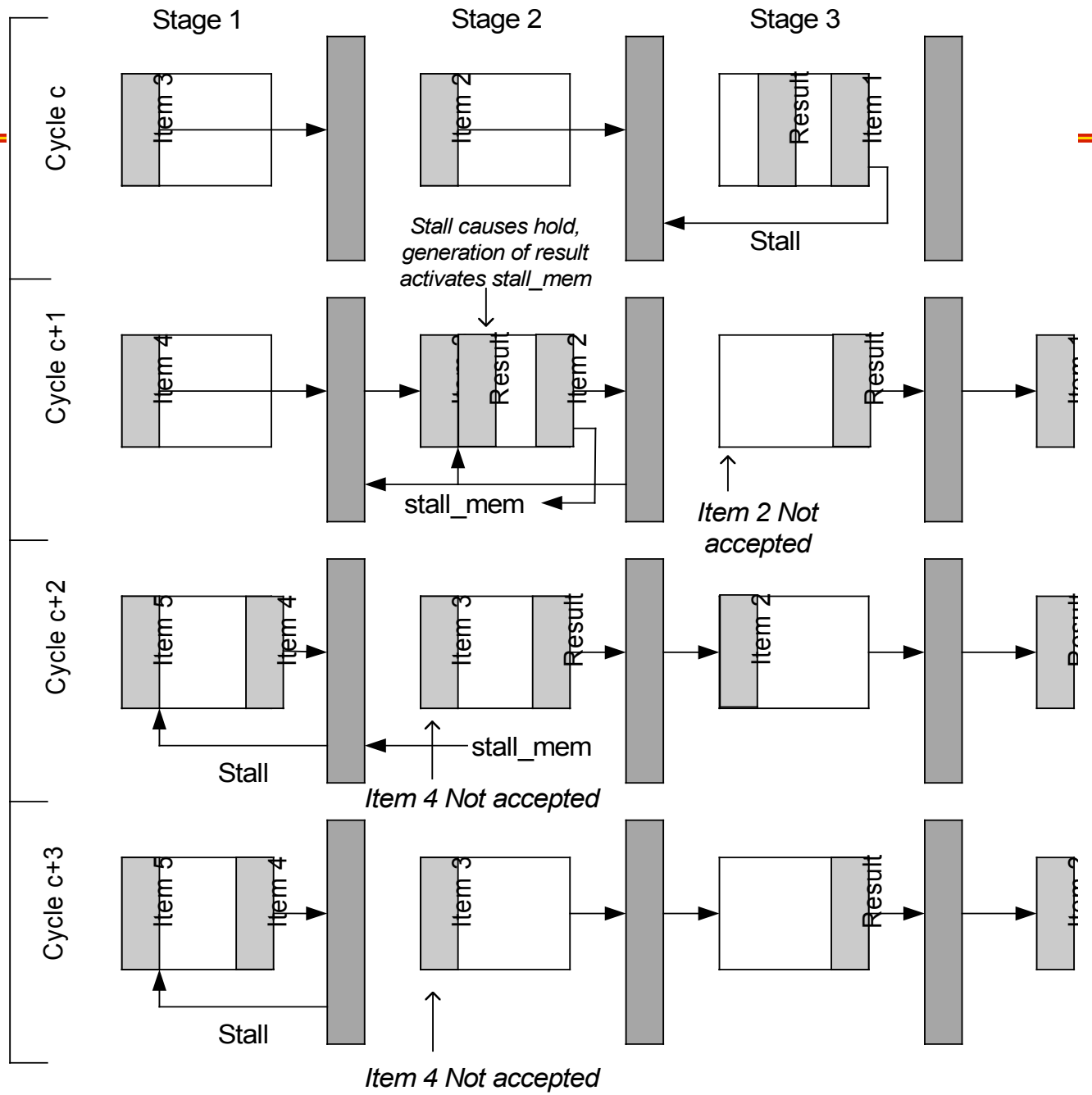
- Uniprocessor Implementations
  - Entire candidate set is entered in a single tree
    - Essentially parallel lookup of all candidates
  - Tree is traversed for all k-subsets of each transaction
    - n = 40, k = 16 --- 40 choose 16 traversals if far more expensive than the 40 cycles required in the single pass hardware case

Systolic Injection

- Key to efficiency
  - add entries to input stream
    - no delay for downstream units
    - upstream units are stalled by only one cycle
      - candidate set is finished, result is appended as suffix
    - Overall number of cycles: number of data elements + number of results

In_{stall}	$stall_{mem}$	Generate	Out_{stall}	$stall_{mem*}$
0	0	0	0	0
0	0	1	1	0
1	0	0	1	0
0	1	d	1	0
1	0	1	1	1
1	1	d	1	1

- If $stall_{mem}$ is active, the system is prevented from generating a result


Stage 1
Stage 2
Stage 3
Cycle c
Cycle c+1
Cycle c+2
Cycle c+3
Item 3
Item 2
Result
Item 1
Stall
Stall causes hold, generation of result activates stall_mem
Item 4
stall_mem
Item 2 Not accepted
Item 5
Item 4 Not accepted

One Stall

0	1	2	3	4	5	6	7	8								
	0	1	2	3	4	5	6	7	8							
		0	1	2	3	4	5	6	7	8						
			0	1	2	3	4	5	6	**7s**	8					
				0	1	2	3	4	**56s**	i	7	8				
					0	1	2	**34s**	5	6	i	7	8			
						0	**12s**	3	4	5	6	i	7	8		
						0s	1	2	3	4	5	6	i	7	8	
							0	1	2	3	4	5	6	i	7	8

Two Stalls Colliding

0	1	2	3	4	5	6	7										
	0	1	2	3	4	5	6	7									
		0	1	2	3	4	5	6	7								
			0	1	2	3	4	5	6	**7s**							
				0	1	2	3	4	**56s**	i	7						
					0	1	2	**34s2**	5	6	i	7					
						0	**12s2**	**i3s**	4	5	6	i	7				
						0s2	**12s**	**3s**	i	4	5	6	i	7			
						0s	1	2	3	i	4	5	6	i	7		
							0	1	2	3	i	4	5	6	i	7	
								0	1	2	3	i	4	5	6	i	7

Bitmapped CAM Architecture (new work to be submitted to IPDPS)

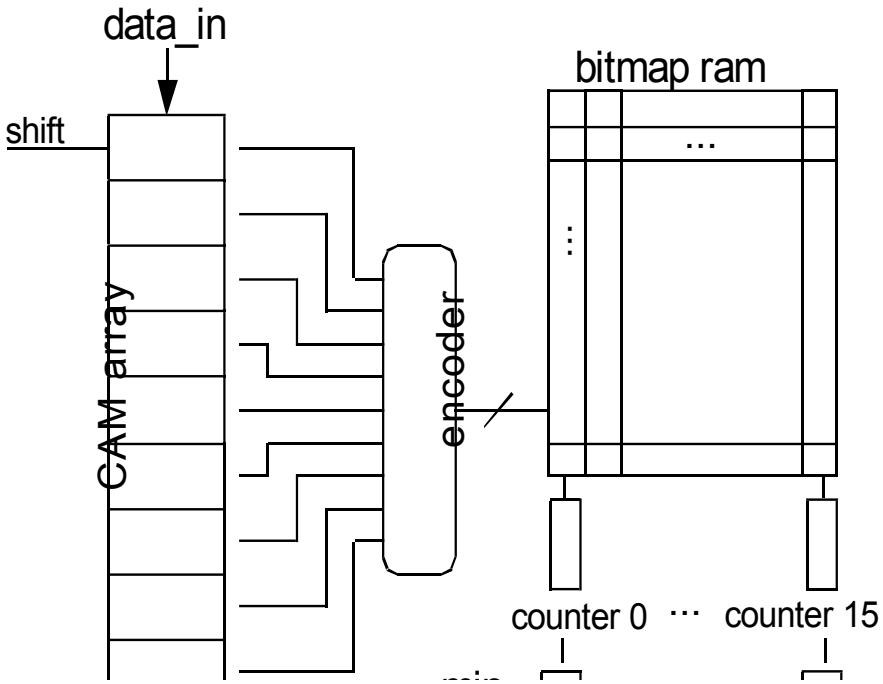
Predecoding in CAM

- Idea from string matching – reduce number of expensive comparators and routing through the grouping of a small number of heavily used elements

249 316 395 482 743 787 819
236 249 395 482 743 787 819
249 316 395 482 743 787 804
236 249 395 482 743 787 804
236 249 316 395 482 743 787
249 319 482 620 743 787 819
249 482 620 743 787 804 819
249 316 482 620 743 787 819
236 249 482 620 743 787 819
249 482 529 620 743 787 819
249 319 482 620 743 787 804

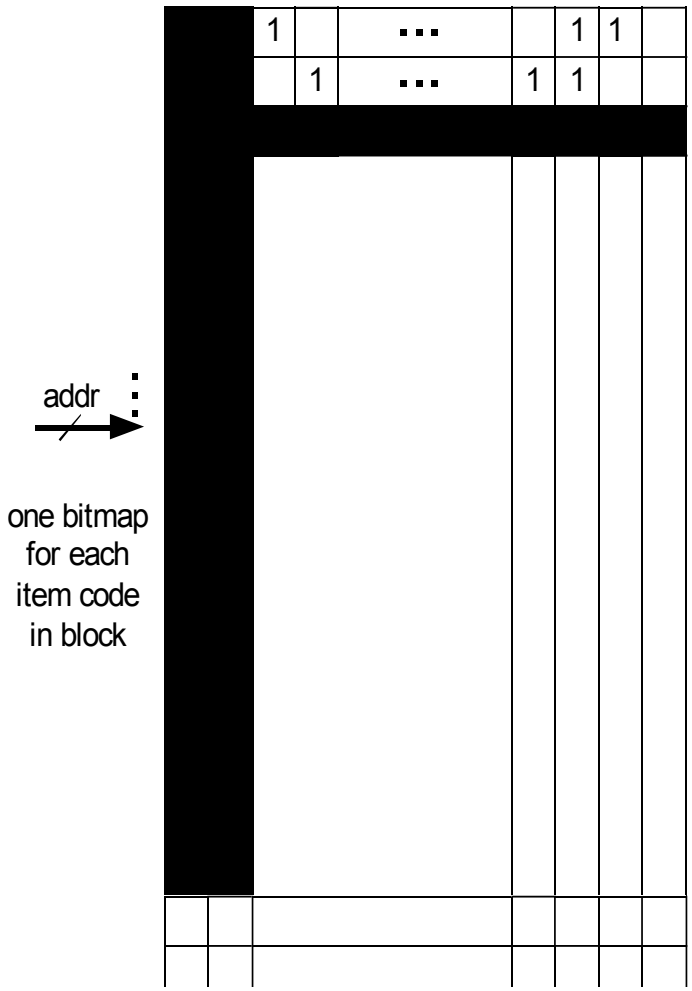
Predecoding Architecture

- CAM entries detect a particular item code
- Bitmap keeps track of which candidates need a given item
- Counters determine if all candidate items have been found



Bitmap Illustration

- Bitmap is computed and optimized by external controller
- Bitmap hold information for 16 candidates and 32 different item codes
  - Number of candidates is reduced if too many item codes are required (rare)



SRC-6 Implementation

- 16 bits per cycle (100 MHz) easy to provide from memory
- Coded in SRC Carte-C
  - Not as much control as VHDL
  - Compiler takes care of the more complicated memory issues
  - Bitmapped CAM blocks replicated across two Virtex 2 6000 –4

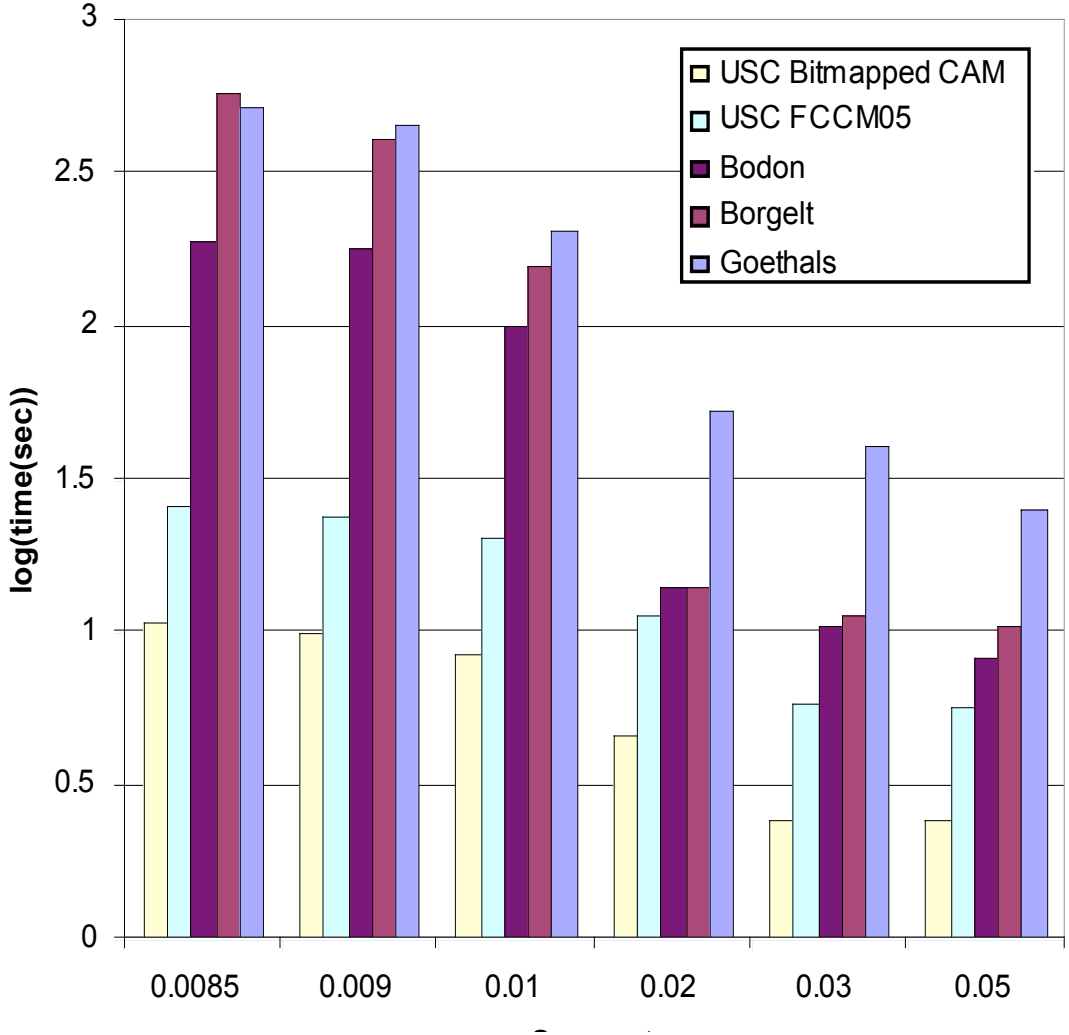
Results

Results

- Based on standard data mining benchmark databases
  - T40I10D100K (15 MB) : average candidate size is 40
  - T10I4D100k (4 MB): average candidate size is 10
- Result comparisons based on tests published in Bodon ‘03

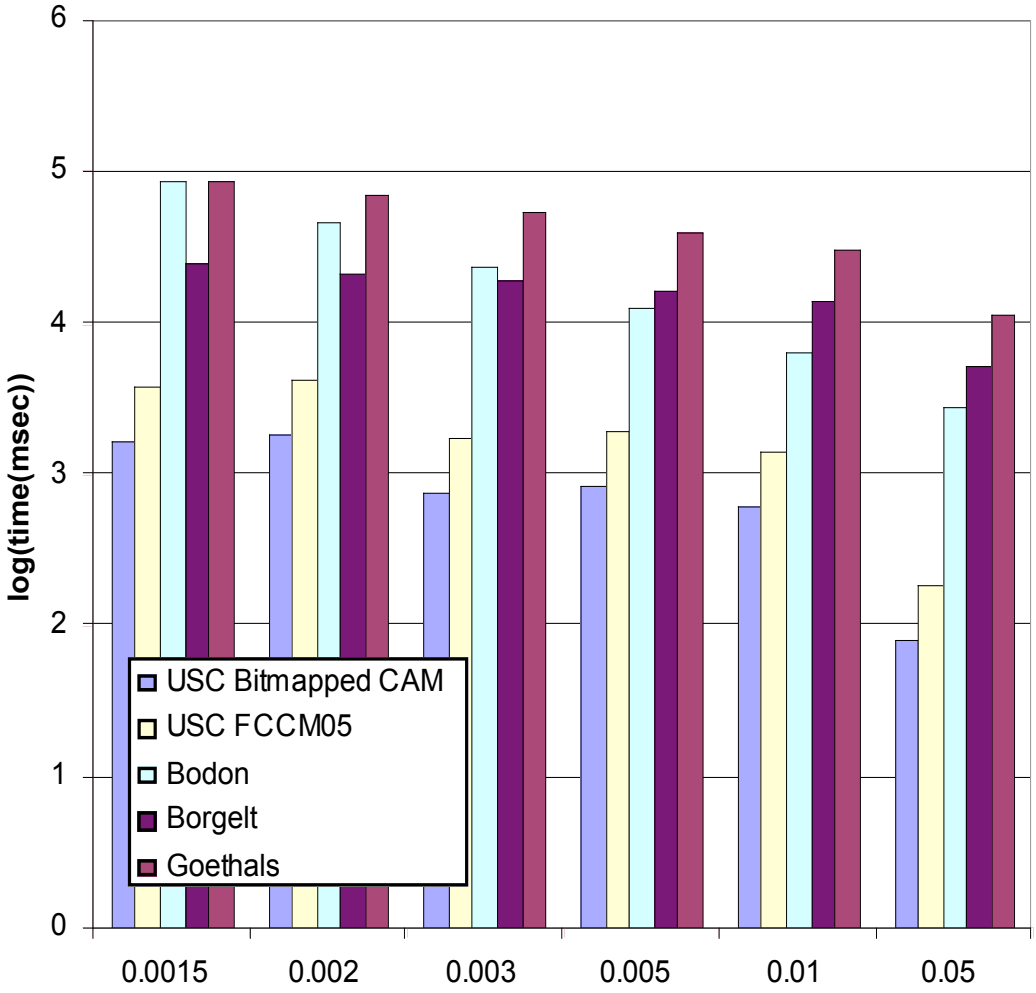
Current Data Mining Results

- T40I10D100K (15 MB) Chart: higher average number of elements in transactions causes more candidates to satisfy support requirements



Initial Datamining Results

- T10I4D100k (4 MB): due to size of database, FPGA running time is essentially proportional to number of candidates



Initial Data Mining Results

- Systolic architecture
  - 560 units on a single V2P100, operating at 125 MHz
  - Number of passes of database is reduced by large number of units
  - Systolic performance (conservative) beats dual Xeon 3-GHz system by 4x (16 MB dataset requires 95 seconds)
- Bitmapped CAM architecture
  - 1400 units on a single V2P100, operating at 135 MHz
  - About 3x higher performance compared to systolic architecture
  - Implemented on SRC6 FPGA accelerated workstation

Future Work

- Port to SRC-7
  - Use of a custom macro in lieu of Carte-C implementation
    - No support for custom bit widths
    - Timing for systolic array requires precise designer control
- Mode-specific reconfiguration for candidate generation and support operations
  - Full application implemented on SRC machine